

Eduardo F. Silva · R. Kevin Wood*

Solving a class of stochastic mixed-integer programs with branch and price

Received: August 26, 2004 / Accepted: August 3, 2005

Published online: April 25, 2006 – © Springer-Verlag 2006

Abstract. We begin this paper by identifying a class of stochastic mixed-integer programs that have column-oriented formulations suitable for solution by a branch-and-price algorithm (B&P). We then survey a number of examples, and use a stochastic facility-location problem (SFLP) for a detailed demonstration of the relevant modeling and solution techniques. Computational results with a scenario representation of uncertain costs, demands and capacities show that B&P can be orders of magnitude faster than solving the standard formulation by branch and bound. We also demonstrate how B&P can solve SFLP exactly – as exactly as a deterministic mixed-integer program – when demands and other parameters can be represented as certain types of independent, random variables, e.g., independent, normal random variables with integer means and variances.

Key words. Stochastic mixed-integer program – Column generation – Branch and price

1. Introduction

This paper describes a class of two-stage stochastic mixed-integer programs (SMIPs) whose instances are amenable to column-oriented formulations, and then shows how to solve such formulations with a branch-and-price algorithm (B&P). The phrase “branch and price” was coined by Savelsbergh [49], but the technique was first proposed by Johnson [33] and implemented by Desrochers and Solomon [25] and Desrochers et al. [24]. Branch and price combines dynamic column generation – this is known widely through the “cutting-stock problem” of Gilmore and Gomory [30] – with standard branch and bound.

Stochastic programmers have only just begun to see that B&P applies to their problems, and we find only two papers on the topic: Damodaran and Wilhelm [21] and Lulli and Sen [42]. (However, Shiina and Birge [51] and Singh et al. [53] use column-generation without branch and bound to solve SMIPs.) Those papers investigate specific

E.F. Silva: Brazilian Navy Analyses Center, Rio de Janeiro, RJ 20091-000, Brazil.
e-mail: efsilva@pobox.com

R.K. Wood: Operations Research Department, Naval Postgraduate School, 1749-016 Lisboa, Monterey, CA 93943, USA. e-mail: kwood@nps.edu

Mathematics Subject Classification (2000): 90C11, 90C15, 47N10

* Kevin Wood thanks the Office of Naval Research, Air Force Office of Scientific Research, the Naval Postgraduate School (NPS) and the University of Auckland for their support. Eduardo Silva thanks NPS and the Brazilian Navy for their support. Both authors are grateful to the COIN-OR team for assistance with computational issues, as well as to two anonymous referees for highly useful, constructive criticism.

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE 26 AUG 2004		2. REPORT TYPE N/A		3. DATES COVERED -	
4. TITLE AND SUBTITLE Solving a class of stochastic mixed-integer programs with branch and price				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Operations Research Department Monterey, CA 93943				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release, distribution unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT We begin this paper by identifying a class of stochastic mixed-integer programs that have column-oriented formulations suitable for solution by a branch-and-price algorithm (B&P). We then survey a number of examples, and use a stochastic facility-location problem (SFLP) for a detailed demonstration of the relevant modeling and solution techniques. Computational results with a scenario representation of uncertain costs, demands and capacities show that B&P can be orders of magnitude faster than solving the standard formulation by branch and bound. We also demonstrate how B&P can solve SFLP exactly as exactly as a deterministic mixed-integer program when demands and other parameters can be represented as certain types of independent, random variables, e.g., independent, normal random variables with integer means and variances.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	18. NUMBER OF PAGES 24	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

applications of B&P to stochastic programming. In contrast, we describe a complete class of problems to which B&P applies; in similarity, we show impressive computational results.

After defining the special class of SMIPs, we provide examples to show how stochastic versions of several well-known deterministic models fit into this framework: the elastic generalized-assignment problem (Brown and Graves [12]), crew-scheduling (Vance et al. [56], Day and Ryan [23]), vehicle-routing problems (e.g., Desrosiers et al. [26]), and the origin-destination integer multicommodity flow problem (Barnhart et al. [6]). One additional problem, a stochastic facility-location problem (SFLP), guides our detailed exploration of the B&P solution approach.

We initially model and solve a version of SFLP with uncertain demands, costs and capacities, all represented through scenarios. Such representations appear frequently in the stochastic-programming literature (e.g., Butler and Dyer [15], Chen et al. [18], Ahmed and Sahinidis [1], Lulli and Sen [42]), with the primary advantage being to allow arbitrary dependence among uncertain parameters. However, another common formulation approach defines individual probability distributions for the stochastic program's parameters, which are typically assumed to be independent (e.g., Bertsimas [10], Zhou and Liu [61]). We will show how B&P can solve SFLP in this situation, too. Furthermore, we will solve the problem exactly, that is, with the same certitude that prevails in deterministic mixed-integer programs. This contrasts with alternative solution procedures that provide only probabilistic or asymptotic guarantees of solution quality (e.g., Carøe and Tind [17], Sen and Hige [50], Ahmed and Sahinidis [1]).

Solution methods for SMIPs with "scenario uncertainty" typically employ Benders decomposition (Benders [9]), or extensions thereof, to the original model. Examples include the integer L-shaped decomposition from Laporte and Louveaux [36], and the methods developed by Carøe and Tind [17] and by Sen and Hige [50]. Unfortunately, all of these decompositions use a master problem whose linear-programming relaxation is no stronger than the linear-programming relaxation of the original model (measured over the master-problem variables which the two formulations have in common). Consequently, these decompositions will suffer if the original model formulation has a poor continuous relaxation. In contrast, B&P solves a column-oriented reformulation of a model, also by a form of decomposition, but that reformulation will normally have a tighter relaxation than the original model (Barnhart et al. [7]).

The next section describes our special class of SMIPs and shows how to convert their standard formulations into column-oriented ones. Several models from the literature provide concrete examples. Section 3 presents the SFLP with scenario uncertainty, describes how to solve instances with B&P, and provides computational results. Section 4 presents a version of SFLP in which random parameters take on continuous distributions, describes how to solve instances with B&P, and provides computational results. Section 5 presents conclusions.

2. General Methodology

We will show that a variety of SMIPs of the following form can arise in scheduling, routing and other applications:

Formulation (SMIP0):

$$\min_{\mathbf{x}} \sum_{i \in I} \left(\mathbf{c}_i \mathbf{x}_i + E_{\tilde{\xi}_i} [h_i(\mathbf{x}_i, \tilde{\xi}_i)] \right) \quad (1)$$

$$\text{s.t. } \sum_{i \in I} A_i \mathbf{x}_i = \mathbf{b} \quad (2)$$

$$\mathbf{x}_i \in X_i \quad \forall i \in I \quad (3)$$

where, for all $i \in I$,

$$h_i(\mathbf{x}_i, \tilde{\xi}_i) = \min_{\mathbf{y}_i} \tilde{\mathbf{f}}_i \mathbf{y}_i \quad (4)$$

$$\text{s.t. } \tilde{D}_i \mathbf{y}_i \geq \tilde{B}_i \mathbf{x}_i + \tilde{\mathbf{d}}_i \quad (5)$$

$$\mathbf{y}_i \in Y_i, \quad (6)$$

and where $\tilde{\xi}_i \equiv \text{vec}(\tilde{B}_i, \tilde{D}_i, \tilde{\mathbf{d}}_i, \tilde{\mathbf{f}}_i)$. The sets X_i require all \mathbf{x}_i to be bounded and integral. The sets Y_i will normally require non-negativity of the \mathbf{y}_i , but we place no particular restrictions on these sets; the variables may be continuous and/or integer. The objective-function term $\sum_{i \in I} E_{\tilde{\xi}_i} [h_i(\mathbf{x}_i, \tilde{\xi}_i)]$ is called the *recourse function* (Walkup and Wets [57]).

We further assume that the model exhibits *relatively complete recourse* (Rockafellar and Wets [47]), which implies that, for any $\mathbf{x}_i \in X_i$, an optimal solution \mathbf{y}_i , satisfying constraints (5) and (6), can always be found. We note that \tilde{D}_i is an identity matrix in some of our examples, implying the property of *simple recourse* (Beale [8], Wets [59]). However, this is not an inherent requirement of this class of problems. The key feature of SMIP0 is that the recourse function decomposes by “subproblem” i .

Because all \mathbf{x}_i are integral and bounded, the principles of Dantzig-Wolfe decomposition apply (Dantzig and Wolfe [22]), as extended to integer programs by Appelgren [4]. (See Wolsey [60], Section 11.2, for a comprehensive discussion.) To describe this decomposition, let $\hat{\mathbf{x}}_i^k \in X_i, k \in K_i$, denote the enumerated vectors satisfying constraints (3) for each i . Because of relatively complete recourse, $E_{\tilde{\xi}_i} [h_i(\hat{\mathbf{x}}_i^k, \tilde{\xi}_i)]$ is well defined for all such $\hat{\mathbf{x}}_i^k$. Then, because of the special structure, we can embed the decomposed recourse function into the column costs of a column-oriented formulation for SMIP0:

Column-Oriented, Mixed-Integer, Two-Stage Stochastic Program (CSMIP0)**Indices:**

$i \in I$ subproblems

$k \in K_i$ indices for (integer) vectors $\hat{\mathbf{x}}_i^k \in X_i$

Data:

$\hat{\mathbf{x}}_i^k$ the k th vector $\hat{\mathbf{x}}_i^k \in X_i$
 \mathbf{c}_i first-stage costs for subproblem i

Decision Variables:

λ_i^k 1 if $\hat{\mathbf{x}}_i^k \in X_i$ is selected as subproblem i 's solution in the overall solution, and 0 otherwise

Formulation (CSMIP0):

$$\min_{\lambda} \sum_{i \in I} \sum_{k \in K_i} \left(\mathbf{c}_i \hat{\mathbf{x}}_i^k + E_{\tilde{\xi}_i} [h(\hat{\mathbf{x}}_i^k, \tilde{\xi}_i)] \right) \lambda_i^k \quad (7)$$

$$\text{s.t.} \sum_{i \in I} \sum_{k \in K_i} \left(A_i \hat{\mathbf{x}}_i^k \right) \lambda_i^k = \mathbf{b} \quad (8)$$

$$\sum_{k \in K_i} \lambda_i^k = 1 \quad \forall i \in I \quad (9)$$

$$\lambda_i^k \in \{0, 1\} \quad \forall i, k \quad (10)$$

Constraints (9) are often referred to as “convexity constraints.” CSMIP0 can be applied when first-stage variables are general integers, but binary variables are typical so we assume this restriction hereafter for simplicity.

Two examples of CSMIP0 to be discussed shortly simplify to the form of a set-partitioning model, so we describe this special case first. If constraints (8) are indexed by $j \in J$, \mathbf{b} is a vector of 1s, A_i is the identity matrix for all i , and all vectors $\hat{\mathbf{x}}_i^k$ are binary, 0–1, then CSMIP0 may be written as:

Formulation (CSMIP1):

$$\min_{\lambda} \sum_{i \in I} \sum_{k \in K_i} \hat{c}_i^k \lambda_i^k \quad (11)$$

$$\text{s.t.} \sum_{i \in I} \sum_{k \in K_i} \hat{x}_{ij}^k \lambda_i^k = 1 \quad \forall j \in J \quad (12)$$

$$\sum_{k \in K_i} \lambda_i^k = 1 \quad \forall i \in I \quad (13)$$

$$\lambda_i^k \in \{0, 1\} \quad \forall i \in I, k \in K_i \quad (14)$$

where $\hat{x}_{ij}^k = (A_i \hat{\mathbf{x}}_i^k)_j$ and $\hat{c}_i^k = \mathbf{c}_i \hat{\mathbf{x}}_i^k + E_{\tilde{\xi}_i} [h(\hat{\mathbf{x}}_i^k, \tilde{\xi}_i)]$.

Naturally, the cardinalities of the index sets K_i in CSMIP0 or CSMIP1 may be enormous, and it will usually be necessary to solve these models without explicitly enumerating the $\hat{\mathbf{x}}_i^k$. Before discussing such issues, however, we will provide concrete examples of how this reformulation technique applies to some stochastic versions of well-known, deterministic optimization problems. We supply only short descriptions of the problems, and ask the reader to check the references for more details. For simplicity, we hereafter drop the subscript on the expectation operator, because it should be clear from context.

2.1. Elastic generalized-assignment problem

(Brown and Graves [12], Appleget and Wood [3])

The objective of the (deterministic) *elastic generalized-assignment problem* (EGAP) is to minimize the cost of assigning capacity-consuming tasks $j \in J$ to capacitated agents $i \in I$, so that (a) each task is assigned to exactly one agent, and (b) the total capacity assigned to agent i does not exceed the agent's (potentially uncertain) capacity \tilde{u}_i , unless an appropriate per-unit penalty f_i is paid. If assigning task j to agent i consumes a random amount of agent i 's capacity, denoted \tilde{b}_{ij} , and if we denote the direct cost of such an assignment as c_{ij} (which could represent an expected value), a stochastic version of the EGAP (SEGAP) can be defined (Spoerl and Wood [54]):

SEGAP

$$\min_{\mathbf{x}} \sum_{i \in I} \left(\sum_{j \in J} c_{ij} x_{ij} + E[h_i(\mathbf{x}_i, (\tilde{\mathbf{b}}_i, \tilde{u}_i))] \right) \quad (15)$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = 1 \quad \forall j \in J \quad (16)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \quad j \in J \quad (17)$$

where

$$h_i(\mathbf{x}_i, (\tilde{\mathbf{b}}_i, \tilde{u}_i)) = \min_{y_i} f_i y_i \quad (18)$$

$$\text{s.t.} \quad y_i \geq \sum_{j \in J} \tilde{b}_{ij} x_{ij} - \tilde{u}_i \quad (19)$$

$$y_i \geq 0. \quad (20)$$

Here, x_{ij} equals 1 if task j is assigned to agent i and is 0 otherwise, and y_i represents capacity violation for agent i . Therefore, $E[h_i(\mathbf{x}_i, (\tilde{\mathbf{b}}_i, \tilde{u}_i))]$ represents the expected capacity-violation penalty for agent i .

The conversion of SEGAP to a column-oriented formulation is straightforward. (Savelsbergh [49] creates the analogous formulation for a deterministic, inelastic GAP.) Each variable represents a potential *joint assignment* of tasks to a particular agent, i.e., a collection of tasks that an agent might be required to perform.

Column-Oriented Formulation for SEGAP (CSEGAP)

Indices:

- $i \in I$ agents
- $j \in J$ tasks
- $k \in K_i$ joint assignments of tasks to agent i

Data:

- \hat{x}_{ij}^k 1 if task j is assigned to agent i in the k th joint assignment for agent i , and 0 otherwise
- $\hat{\mathbf{x}}_i^k$ $(\hat{x}_{ij_1}^k, \hat{x}_{ij_2}^k, \dots, \hat{x}_{ij_{|J|}}^k)$, the k th joint-assignment vector for agent i
- X_i the set of all possible joint assignments $\hat{\mathbf{x}}_i^k$ for agent i (the index set K_i can now be defined as the minimal set such that $\cup_{k \in K_i} \hat{\mathbf{x}}_i^k = X_i$)
- \hat{c}_i^k expected cost of the k th joint assignment for agent i ($\hat{c}_i^k = \mathbf{c}_i \hat{\mathbf{x}}_i^k + E[h_i(\hat{\mathbf{x}}_i^k, (\tilde{\mathbf{b}}_i, \tilde{u}_i))]$ for all $i \in I$ and $k \in K_i$)

Decision Variables:

- λ_i^k 1 if the k th joint assignment is selected for agent i , and 0 otherwise

Formulation (CSEGAP): Same as CSMIP1, Equations (11)–(14).

Constraints (12) guarantee that each task j is assigned to exactly one agent, and constraints (13) ensure each agent i receives exactly one joint assignment of tasks. Note that this simple model allows any group of tasks to be assigned to any agent, so X_i consists of all binary vectors of length $|J|$, implying that $|K_i| = 2^{|J|}$ for all i . This model can, indeed, possess a large number of columns.

2.2. Routing and scheduling with time windows (Desrosiers et al. [26], Ribeiro and Soumis [46])

The *vehicle routing problem with time windows* (VRPTW) is one important exemplar from our special problem class. VRPTW describes a fleet of vehicles that must deliver a set of customer orders, with each customer being represented by a node in a network. Vehicles have limited capacities, and customers specify windows of time during which deliveries should be made. The model allocates orders to vehicles so that vehicle capacities are respected, and identifies a route for each vehicle that delivers each order during that order's customer-specified time window. The column-oriented formulation for this problem fits the form of CSMIP1, Equations (11)–(14), with indices, parameters and variables appropriately defined. The parameters $\hat{\mathbf{x}}_i^k, k \in K_i$, now represent potential routes and sets of deliveries to customers for vehicle i , each of which covers a subset of the customer set J . A probabilistic recourse function could include expected penalties for violating time windows and expected penalties for exceeding maximum route duration.

2.3. Crew scheduling

(Vance et al. [56], Day and Ryan [23])

Following the description in [56], an airline crew-scheduler wishes to minimize the cost of assigning flight crews to a fixed schedule of flights. *Crew pairings* define feasible trip itineraries that can be assigned to some crew. Each pairing consists of a sequence of flights that starts and ends at a home base, respects limits on work hours, allows times for rest breaks, and satisfies numerous other requirements. Set-partitioning models are the norm for this type of problem (e.g., [56], [23]), and these fit a simplified form of CSMIP0 in which the set I is a singleton, \mathbf{b} is a vector of 1s corresponding to flights that must be covered by crews, and the convexity constraints (9) are eliminated. The binary columns $A_i \hat{\mathbf{x}}_i^k$ represent pairings.

However, “home-base constraints” may need to be enforced (e.g., Butchers et al. [14]) and these simply modify constraints (9) to

$$\sum_{k \in K_i} \lambda_i^k \leq u_i \quad \forall i \in I, \quad (21)$$

where I denotes the set of home bases, u_i denotes the number of crews available at $i \in I$, and the index sets K_i now represents potential pairings for crews based at i . For the recourse function, we suggest a probabilistic variant on the function that Ehrgott and Ryan [28] use to penalize schedules that do not allow adequate time for crews to switch aircraft. Their function is based on averaged historical information, but could be modified to represent a penalty function integrated over empirical or fitted delay distributions.

2.4. Origin-destination integer multi-commodity flow problem

(Barnhart et al. [6])

The origin-destination integer multi-commodity flow problem restricts the standard, linear multi-commodity flow problem (Ahuja et al. [2], chapter 17) by requiring each of a set of commodities to be shipped from its origin to its destination along a single path. This problem’s formulation resembles the well-known path-oriented (i.e., column-oriented) formulation of the linear multi-commodity flow problem (Ford and Fulkerson [29]), but with binary variables. In particular, $\lambda_i^k = 1$ if commodity i follows path $k \in K_i$, and $\lambda_i^k = 0$ otherwise. The main constraints of this problem require that the sum of all commodities flowing across each arc respect that arc’s capacity. Constraints (8), converted to inequalities, handle these requirements if we (a) let b_j represent the capacity of arc j , and (b) define $a_{ij} \hat{x}_i^k$ to be the amount of arc j ’s capacity consumed if commodity i is shipped using path k . The convexity constraints (9) guarantee selection of a single path, with appropriate origin and destination, for each commodity. For a communications network, say, each component of the recourse function $E[h_i(\mathbf{x}_i, \xi_i)]$ might represent an expected, path-dependent penalty based on uncertain link availability (Girard and Sansó [31]) or uncertain “hop delay” that is independent of congestion (Papagiannaki et al. [44]).

The following section investigates, in detail, one additional problem that fits the framework of SMIP0, CSMIP0, and more specifically, CSMIP1.

3. Solving a stochastic facility location problem by branch and price

3.1. A stochastic facility location problem with sole sourcing

A standard, deterministic, facility-location problem seeks the best locations at which to open, i.e., build or lease, capacitated production facilities that will ship to established customers to meet those customers' demands for some product. (We consider only the simplest, single-product case here.) The mathematical model must find the best trade-off between variable and fixed costs (Laporte et al. [36]): Opening more facilities leads to lower shipping (variable) costs because plants are closer to customers, on average, but opening more facilities incurs more facility-installation (fixed) costs. The deterministic model typically assumes that all customer demands will be completely satisfied, and sometimes requires that each customer be served by a unique facility. This latter requirement is known as *sole-sourcing*, and the resulting model is called the (deterministic) capacitated facility-location problem with sole-sourcing (FLP) (Barcelo and Casanova [5]).

Assume now that some uncertainty arises in the nominally deterministic FLP: Does a manufacturer really know what future costs, demands and capacities will be? Let us, initially, represent this uncertainty through a finite, discrete set of scenarios indexed by s , with probability of occurrence equaling p^s , and with \mathbf{c}^s , \mathbf{u}^s , \mathbf{d}^s and \mathbf{f}^s representing corresponding shipping costs, facility capacities, customer demands and "excess-demand" penalties, respectively. For simplicity, we assume that if the aggregate demand for a facility exceeds its capacity to produce, the facility pays a penalty based on the excess. Thus, the model that follows will be reasonable if, when a facility runs short of supply, it purchases extra product from an outside supplier and actually satisfies the excess demand by shipping this product appropriately.

Stochastic Facility Location Problem with Sole-Sourcing (SFLP)

Indices:

$i \in I$ potential facilities (at various locations)

$j \in J$ customers

$s \in S$ scenarios

Data [units]:

c'	fixed cost to open facility i [dollars]
d_j^s	customer j 's demand under scenario s [tons]
u_i^s	facility i 's capacity under scenario s [tons]
f_i^s	penalty for each unit of excess demand at facility i under scenario s [dollars/ton]
p^s	probability that scenario s occurs
c_{ij}^s	per-unit shipping cost from i to j under scenario s [\$/ton]
\bar{c}_{ij}	expected cost to supply all of customer j 's demand from facility i , disregarding any shortfalls in facility i 's capacity [dollars] ($\bar{c}_{ij} = \sum_{s \in S} p^s c_{ij}^s d_j^s$.)

Decision Variables [units]:

- x'_i 1 if facility i is opened, and 0 otherwise
 x_{ij} 1 if customer j is assigned to facility i , and 0 otherwise
 y_i^s amount of excess demand at facility i under scenario s [tons]

Formulation (SFLP):

$$\min_{\mathbf{x}', \mathbf{x}, \mathbf{y}} \sum_{i \in I} c'_i x'_i + \sum_{i \in I} \sum_{j \in J} \bar{c}_{ij} x_{ij} + \sum_{s \in S} \sum_{i \in I} p^s f_i^s y_i^s \quad (22)$$

$$\text{s.t. } -x'_i + x_{ij} \leq 0 \quad \forall i \in I, j \in J \quad (23)$$

$$\sum_{i \in I} x_{ij} = 1 \quad \forall j \in J \quad (24)$$

$$\sum_{j \in J} d_j^s x_{ij} - y_i^s \leq u_i^s \quad \forall i \in I, s \in S \quad (25)$$

$$x'_i \in \{0, 1\} \quad \forall i \in I \quad (26)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J \quad (27)$$

$$y_i^s \geq 0 \quad \forall i \in I, s \in S \quad (28)$$

This type of formulation is known as the *extensive form* of a stochastic program (Birge and Louveaux [11], p. 8), because the second-stage variables and constraints are made explicit for all scenarios.

3.2. A column-oriented formulation for SFLP

Here we describe CSFLP, a column-oriented formulation for SFLP that fits directly into the form of CSMIP1. In this formulation, a *joint assignment* represents any collection of customers that are served by the same facility. We note that Teo and Shu [55] and Lorena and Senne [38] have previously used column generation for solving deterministic facility-location problems. Both of these papers use master problems that resemble our set-partitioning master problem. The main differences lie in the subproblems: Theirs are deterministic while ours is stochastic.

Column-Oriented Formulation of SFLP (CSFLP)**Indices:**

- $k \in K_i$ possible joint assignments of customers to facility i

Data [units]:

\hat{x}_{ij}^k 1 if customer j is assigned to facility i in the k th joint assignment for that facility, and 0 otherwise
 \hat{c}_i^k total expected cost of the k th joint assignment for facility i ($\hat{c}_i^k = c'_i + \sum_{j \in J} \bar{c}_{ij} \hat{x}_{ij}^k + \sum_s p_s f_i^s \hat{y}_i^s$, except $\hat{c}_i^k = 0$ for the null assignment) [dollars]

Decision Variables:

λ_i^k 1 if the k th joint assignment of customers to facility i is selected, and 0 otherwise

Formulation (CSFLP): Same as CSMIP1, Equations (11)–(14).

A column-oriented formulation like CSFLP cannot be solved directly because it is impossible, or impractical, to create the full set of columns. Therefore, each set K_i is replaced by a subset to form a *restricted master problem* (RMP). The solution to the LP relaxation of the RMP (LP-RMP) then yields dual variables, which can be used to attempt to identify one or more new columns with favorable reduced costs through one or more *column-generation subproblems*. In the case of CSFLP, if we seed the RMP with all null joint assignments, the following subproblem arises for each facility i :

CSUB _{i} ($\hat{\pi}$, $\hat{\mu}_i$)

$$z_i^* = \min_{\mathbf{x}_i, \mathbf{y}_i} \sum_{j \in J} (\bar{c}_{ij} - \hat{\pi}_j) x_{ij} + \sum_{s \in S} p^s f_i^s y_i^s + c'_i - \hat{\mu}_i \quad (29)$$

$$\text{s.t.} \quad \sum_{j \in J} d_j^s x_{ij} - y_i^s \leq u_i^s \quad \forall s \in S \quad (30)$$

$$x_{ij} \in \{0, 1\} \quad \forall j \in J \quad (31)$$

$$y_i^s \geq 0 \quad \forall s \in S, \quad (32)$$

where $\hat{\pi}_j$ is the optimal dual variable associated with constraint (12) for customer j in LP-RMP, and $\hat{\mu}_i$ is the optimal dual variable from LP-RMP for the convexity constraint (13) associated with facility i . If the per-unit shipping cost from facility i to customer j is a random quantity, denoted \tilde{c}_{ij} , note that $\bar{c}_{ij} = E[\tilde{c}_{ij} \tilde{d}_j]$, and $\tilde{c}_{ij} = E[\tilde{c}_{ij}]E[\tilde{d}_j]$ if independence of \tilde{c}_{ij} and \tilde{d}_j prevails.

If the solution to CSUB _{i} ($\hat{\pi}$, $\hat{\mu}_i$) defines a non-null joint assignment of customers to facility i , z_i^* gives the reduced cost of the joint assignment with respect to LP-RMP's current solution. A negative reduced cost indicates that $\hat{\mathbf{x}}_i^k$ should be translated into a column for the RMP, and inserted into it, and a non-negative reduced cost indicates that no favorable column currently exists for facility i . (A positive reduced cost for subproblem i can arise if K_i contains only the null schedule.)

3.3. Solving the column-generation subproblems

The subproblems $\text{CSUB}_i(\hat{\pi}, \hat{\mu}_i)$ are *multi-dimensional knapsack problems* (Weingartner and Ness [58]) with elastic penalties in each dimension; Kleywegt et al. [34] refer to these as *static stochastic knapsack problems*. We solve them through straightforward branch and bound, except that we add “explicit constraint branching” (Appleget and Wood [3]) by defining the general integer variables g_i and adding the following constraint to each subproblem i :

$$\sum_{j \in J} x_{ij} - g_i = 0. \quad (33)$$

The variable g_i is an “ECB variable” and receives a higher priority for branching than does any x_{ij} . Intuitively, constraint branching provides a better balanced branch-and-bound enumeration tree, and this tends to reduce enumeration (see Ryan and Foster [48]).

3.4. Solving the LP-relaxation of the master problem

Branch-and-price algorithms (e.g., Savelsbergh [49], Barnhart et al. [7], Silva [52]) are appearing as complements to the branch-and-cut algorithms that are commonly implemented in commercial MIP solvers. B&P combines a branch-and-bound algorithm with a column-generation procedure. Achieving good performance with column-generation is difficult (Lübbecke and Desrosiers [41]), but a number of enhancements to the basic procedure can help. “Duals stabilization” comprises the most important enhancement in our experience, so we describe that here briefly. (See du Merle et al. [27] and Silva [52] for more detail.)

Duals stabilization attempts to accelerate the column-generation process that solves CSFLP’s LP relaxation. We follow du Merle et al. [27] for this purpose, and incorporate an *elastic dynamic trust region* for dual variables. The trust region is always centered on the most recent dual solution. It is elastic because penalized violation of the nominal trust region is allowed, and it is dynamic because its width and penalties are adjusted continually. This trust-region mechanism is implemented by turning master-problem equality constraints into elastic ranged constraints. The primal (master-problem) elastic penalties define the dual trust region’s limits, while primal ranges define the dual penalties, i.e., the penalties applied if the dual variables fall outside the nominal trust region.

A trust region of some sort makes sense in this context because (a) the column-generation mechanism, when viewed in the dual, is essentially Benders decomposition (Benders [9]), and (b) the Benders master problem appears to benefit from the use of trust regions (e.g., Brown et al. [13], Linderoth and Wright [37]). Of course, many variants on trust regions could be applied to our problems, but this one is simple and has proven effective in recent column-generation experiments (Silva [52], Singh et al. [53]). (Actually, experiments in [53] suggest an even better method for “duals stabilization,” one which we have not tested: Simply use interior-point duals as provided by an interior-point solution of LP-RMP.)

3.5. Computational results

We implement B&P using software from the COmputational INfrastructure for Operations Research (“COIN-OR,” or simply “COIN”), which provides a repository of distinct libraries that can be integrated to build optimization algorithms (Lougee-Heimer [39]). The COIN library labeled “BCP” provides the basic framework for a B&P algorithm (Ralphs and Ladanyi [45]). This library’s design anticipates a parallel/distributed environment, and, unfortunately, the protocol that emulates this environment in our serial environment incurs some computational overhead. This overhead could be avoided with some additional programming, so the total solution times reported here, denoted (TT), exclude that overhead. However, we note that the true CPU times for our implementations never exceed TT by more than 10%, and the mean overhead for all problems is only 3.1%.

We have implemented our B&P algorithm using COIN’s open solver interface (OSI), coupled with CPLEX 8.0. The linear relaxation of the RMP and the subproblems are submitted to CPLEX’s LP solver and MIP solver, respectively. We carry out these tests on a networked workstation, a Dell Dimension 340 with a 2 GHz, Pentium IV processor and 1 GB of RAM. For comparison, we also directly solve the extensive formulations of SFLP using CPLEX 8.0, and report these solution times under “IP” in the Tables 1 and 2.

We note that the binary status of the variables x_{ij} in the extensive formulation would force continuous versions of the variables x'_i to be binary in any optimal solution. Consequently, we need not indicate to the solver that the x'_i are binary. However, we find that solution times are shorter, on average, when we specify the x'_i to be binary and set the branching priorities on these variables to be higher than those on the x_{ij} . Thus, we force branching first on the important decisions, i.e., whether or not to open a plant, rather than on the relatively less important, and more numerous, customer-to-plant assignment decisions.

We investigate eight groups of problems. Each group is defined by problem size, meaning “number of facilities-number of customers,” and these sizes are: 5–15, 5–30, 8–24, 8–48, 10–30, 10–40, 10–50 and 10–60. For each problem size, we consider instances with 1, 10 or 50 scenarios. (We consider larger problems with more scenarios later.) Because run times vary somewhat between randomly generated instances of the same size, we examine five different instances for each combination of problem size and number of scenarios. All problems in this paper are solved to optimality.

To generate the test problems, we first create a reference problem – the superscript “ R ” below stands for “reference” – according to the following rules: (a) Customer demands d_j^R are integers taken from a discrete uniform distribution $U(5,25)$, (b) transportation costs c_{ij}^R are integers from $U(15,25)$, (c) facility capacities are $u_i^R = 0.8 \sum_{j \in J} d_j^R / |I|$, and (d) the fixed costs are $c'_i = \rho u_i^R$ for some cost-per-unit-capacity conversion constant ρ , which is 1.5 for these examples. Chu and Beasley [19] use rules (a) and (b) to generate certain instances of the generalized-assignment problem. For our stochastic instances, demands d_j^s are uniformly distributed integers within $\pm 20\%$ of d_j^R , capacities u_i^s are $\pm 10\%$ of u_i^R , and fixed costs are simply $c'_i = c'^R_i$. Also, facility i pays an additional $f_i^s = 0.4 \max_{j \in J} c_{ij}^s$ dollars for each unit of demand it must satisfy through an

Table 1. Total time (TT) in CPU seconds, to solve randomly generated SFLPs with scenario uncertainty. Three different algorithms solve five problem instances for each combination of size and number of scenarios. (Note: All generated scenario data for the problem instance in row r , for $r = 1, \dots, 10$, have been reused in row $r + 5$. This accounts for apparent correlations in runtimes as exemplified by rows 5, 10 and 15.) A time given in the bold font indicates the fastest among the three alternative solution methods

Number of Scenarios	Problem Size (facilities-customers)											
	5–15			5–30			8–24			8–48		
	B&P		B&P	B&P		B&P	B&P		B&P	B&P		B&P
	IP	w/o Stz	w/ Stz	IP	w/o Stz	w/ Stz	IP	w/o Stz	w/ Stz	IP	w/o Stz	w/ Stz
1	3.5	0.5	0.8	2.5	1.7	1.7	2274.9	1.1	14.5	5.8	5.0	2.7
1	0.1	0.3	0.8	1.5	4.1	3.9	*	2.0	2.7	3.8	3.4	2.4
1	3.2	0.6	2.0	0.9	1.4	1.9	274.1	1.6	1.8	4.5	6.4	2.7
1	0.2	0.3	0.6	1.6	2.1	1.9	299.8	1.3	4.4	3.8	3.2	2.6
1	3.6	0.5	1.4	8.6	2.4	2.8	1.8	0.8	0.9	5076.1	61.2	44.6
10	1.3	1.1	1.6	4.2	3.7	3.7	881.0	4.8	9.1	142.4	7.9	4.2
10	1.4	1.0	1.4	5.1	16.6	30.5	2987.5	4.3	7.0	39.7	6.7	5.7
10	1.0	0.9	1.1	1.2	2.8	3.0	231.3	4.8	5.7	20.6	8.4	5.2
10	0.4	0.6	1.3	2.4	3.2	3.6	16.4	2.7	2.9	19.8	6.7	5.9
10	1.0	0.7	1.4	9.4	3.6	6.8	3.9	1.2	1.3	*	29.7	25.8
50	2.5	2.3	3.0	4.9	10.1	10.2	1637.2	45.5	29.6	455.6	40.0	58.5
50	3.1	2.3	4.4	11.0	11.1	11.1	5579.0	8.7	7.6	45.7	20.2	14.0
50	2.8	2.3	2.7	3.4	7.8	8.5	1081.6	8.0	7.6	53.4	25.0	15.9
50	1.3	1.4	3.1	4.3	8.6	10.1	57.4	5.8	6.0	129.1	24.3	20.9
50	3.9	2.7	3.3	12.2	12.1	10.7	6.6	4.2	4.2	990.2	80.1	114.6

Legend:

IP: CPLEX MIP solver solving the extensive-form SFLP

B&P w/o Stz: Branch-and-price algorithm without duals stabilization solving CSFLP

B&P w Stz: Branch-and-price algorithm with duals stabilization solving CSFLP

* Problem not solved to optimality in 7,200 CPU seconds.

outside purchase. Clearly, these parameter settings are somewhat arbitrary, but testing suggests that, over a wide range of settings, B&P remains a superior method for solving these problems, compared to solving them directly. As evidence, the reader will see in Tables 1 and 2 that B&P can even solve single-scenario problems faster than they can be solved directly. That is, B&P can be faster even when parameter variances are 0. Furthermore, experiments with $|S| > 1$ not reported here indicate that B&P's superiority only increases as parameter variances increase beyond the values used in this paper.

Tables 1 and 2 show TT for each problem instance solved (a) in its extensive form by IP, (b) as CSFLP by basic B&P without duals stabilization, and (c) as CSFLP by B&P with duals stabilization. Parameter settings with duals stabilization are fixed for all problems tested, and we set an arbitrary limit of 7,200 seconds on total allowed computation time.

3.6. Discussion

Both Tables 1 and 2 provide stark evidence that branch and price can be vastly superior to branch and bound for solving certain SMIPs, and even for certain deterministic MIPs as indicated by the results for the single-scenario problems. One key to this superiority

Table 2. Total time (TT) in CPU seconds, to solve randomly generated SFLPs with scenario uncertainty. Three different algorithms solve five problem instances for each combination of size and number of scenarios. This table explores how computation times change as the ratio of facilities to customers decreases. A time given in the bold font indicates the fastest among the three alternative solution methods

Number of Scenarios	Problem Size (facilities-customers)											
	10–30			10–40			10–50			10–60		
	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz
1	*	16.7	17.0	15.1	3.0	2.2	*	67.5	143.0	21.5	6.8	3.8
1	3.5	1.1	1.1	7.8	2.1	2.1	*	53.8	50.8	22.3	11.0	5.3
1	44.8	1.0	1.5	14.8	2.8	2.1	1657	9.7	7.2	14.8	8.3	4.4
1	1.4	1.3	1.5	*	36.2	10.2	*	116.8	99.1	13.5	11.9	5.2
1	*	5.3	19.5	6.2	3.2	1.7	50	5.5	4.1	23.8	13.4	4.3
10	*	10.3	12.9	2323.4	6.0	4.0	*	20.6	13.9	37.2	16.7	8.0
10	5.2	2.4	1.8	860.1	5.2	3.8	*	310.4	424.2	3163.1	30.2	22.4
10	7.0	3.3	2.6	262.6	5.6	4.1	*	18.0	14.1	140.5	23.1	14.5
10	5.0	2.0	2.1	*	53.6	77.2	*	19.7	11.7	45.6	14.2	10.8
10	*	15.0	29.4	613.5	5.1	11.2	*	20.5	41.2	30.7	19.7	9.6
50	*	16.2	15.6	3347.3	20.6	14.2	*	99.1	127.2	154.6	37.7	18.8
50	12.6	6.5	7.1	1894.9	14.5	11.6	*	37.9	22.9	*	60.0	50.7
50	51.7	7.7	10.7	573.8	14.8	11.3	*	112.3	171.7	132.3	47.9	39.9
50	16.6	6.7	6.0	*	30.8	27.3	*	33.9	27.9	97.1	34.4	21.5
50	*	26.0	115.0	3559.3	17.3	12.3	*	72.7	111.5	213.3	39.7	22.2

Legend: Same as Table 1

clearly lies in the tighter LP lower bound provided by the column-oriented formulation versus the extensive formulation: see Tables 3 and 4.

One might be concerned that B&P requires so much overhead that it could not be effective for small problems. However, the problems in Table 1 have only five or eight potential facilities, and IP outperforms B&P only in problems with five facilities, and then only by a small amount. Moreover, average solution times for B&P are at least an order of magnitude faster than IP, and IP cannot even solve two of the problems within the time limit of 7,200 CPU seconds. Even for small problems, B&P is a good choice.

Table 2, which covers problems with 10 facilities and 30 to 60 customers, clearly shows that B&P solution times are more stable and suffer less than IP when the number of scenarios increases. Observe that (a) 23 problem instances out of 60 could not be solved by IP within 7,200 CPU seconds, (b) IP never outperforms B&P, and (c) B&P can be orders of magnitude faster than solving the original problem, even for single-scenario instances, i.e., for deterministic problems.

Table 5 explores the computational limits of our current B&P implementation by covering a wider range of problem sizes and number of samples than do Tables 1 and 2. Camm et al. [16] solve a facility-location model for a commercial application with 17 potential facilities and 123 customer zones, so our largest problem is roughly the same size as at least one real-world problem. We can see here (and to a degree in Tables 1 and 2) that solution times tend to increase only slowly, perhaps linearly, with the number of scenarios. Thus, the number of scenarios does not seem to be a strongly limiting factor with the B&P methodology. This bodes well for solving an SMIP through sampled approximating problems, since the probability of identifying the optimal solution

Table 3. Integrality gaps compared between the extensive formulation of SFLP (SFLP) and the column-oriented formulation (CSFLP). These results correspond to the problems in Table 1

Number of Scenarios	Problem Size (facilities-customers)							
	5–15		5–30		8–24		8–48	
	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)
1	18.64	0.00	18.15	0.00	25.29	0.00	22.18	0.00
1	28.48	0.00	22.33	0.00	27.25	0.05	23.26	0.00
1	48.20	0.00	26.65	0.00	27.00	0.00	24.67	0.00
1	16.78	0.04	18.43	0.00	34.65	0.00	22.65	0.00
1	22.16	0.00	19.26	0.00	22.27	0.00	22.49	0.05
10	19.21	0.03	17.11	0.00	25.93	0.07	23.28	0.00
10	34.97	0.00	23.14	0.04	26.21	0.02	21.50	0.00
10	37.36	0.00	24.38	0.00	28.52	0.00	23.79	0.03
10	18.62	0.00	17.77	0.00	31.37	0.00	23.29	0.00
10	20.09	0.00	19.47	0.00	20.93	0.00	21.13	0.00
50	19.37	0.00	17.05	0.00	24.75	0.07	23.66	0.01
50	37.02	0.00	23.15	0.00	25.90	0.00	21.39	0.00
50	36.55	0.00	24.15	0.00	28.68	0.00	24.03	0.02
50	18.87	0.00	17.86	0.00	30.04	0.00	23.73	0.00
50	20.33	0.00	19.22	0.00	21.05	0.00	20.57	0.02

for a discrete stochastic program increases exponentially with the number of sampled scenarios (Kleywegt et al. [34]).

Table 5 also shows that problem size, in terms of facilities and customers, is a stronger limiting factor in solving SFLP by B&P. We discuss potential reasons for this in the following sections.

4. Solving a special case of SFLP exactly

Here we investigate a special case of the SFLP in which uncertain parameters are continuously distributed random variables satisfying certain independence requirements. This model paradigm appears frequently in the literature (e.g., Louveaux and Peeters [40], Laporte et al. [36]); however, such models are rarely solved exactly as we shall solve SFLP. Even if the reader believes that some of our independence assumptions are unreasonable in a real-world facility-location problem – we require independence of demands, for instance, and this seems particularly unlikely in the SFLP – it is instructive to see that exact solutions can be achieved for such a model in the column-oriented framework. Perhaps the independence assumptions will be more appropriate in other applications.

Consider a random vector $\tilde{\xi} = \text{vec}(\tilde{\mathbf{c}}, \tilde{\mathbf{d}}, \tilde{\mathbf{f}})$ whose elements represent shipping costs, customer demands and excess-demand penalties, respectively; facility capacities \mathbf{u} will be deterministic initially. Using this definition of $\tilde{\xi}$, and the following detailed definitions, the column-oriented formulation for SFLP clearly fits the form of CSMIP1.

Data [units]

- c'_i fixed cost for opening facility i [dollars]
- \tilde{c}_{ij} per-unit shipping cost from facility i to customer j [dollars/ton]

Table 4. Integrality gaps compared between the compact formulation of SFLP (SFLP) and the column-oriented formulation (CSFLP). These results correspond to the problems in Table 2

Number of Scenarios	Problem Size (facilities-customers)							
	10–30		10–40		10–50		10–60	
	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)
1	22.35	0.04	32.07	0.00	40.43	0.07	20.72	0.00
1	23.00	0.00	26.59	0.00	41.53	0.06	32.15	0.00
1	21.04	0.00	26.16	0.00	56.96	0.02	25.89	0.00
1	23.92	0.00	26.59	0.00	52.95	0.10	23.19	0.00
1	38.46	0.08	27.41	0.00	45.97	0.00	22.71	0.00
10	22.69	0.02	32.30	0.00	37.85	0.02	21.04	0.00
10	22.28	0.00	27.02	0.00	38.12	0.12	35.53	0.01
10	20.32	0.00	25.77	0.03	55.77	0.00	27.42	0.02
10	21.80	0.00	27.55	0.05	48.76	0.04	23.96	0.00
10	42.51	0.09	27.05	0.00	46.41	0.03	21.74	0.00
50	22.66	0.00	33.43	0.00	38.46	0.03	20.21	0.00
50	21.65	0.00	27.64	0.00	36.00	0.00	35.39	0.00
50	20.73	0.00	25.55	0.00	55.82	0.09	26.89	0.00
50	21.82	0.00	28.18	0.00	47.38	0.00	24.36	0.00
50	42.30	0.06	28.03	0.00	44.95	0.06	22.23	0.00

\tilde{d}_j demand from customer j [tons]

\tilde{c}_{ij} expected cost to supply all of customer j 's demand from facility i , disregarding any shortfalls in facility i 's capacity, i.e., $\tilde{c}_{ij} = E[\tilde{c}_{ij}\tilde{d}_j]$ [dollars]

u_i capacity of facility i [tons]

\tilde{f}_i per-unit penalty for excess demand that must be covered by facility i [dollars/tons]

\hat{c}_i^k total expected cost of the k th joint assignment of customers to facility i [dollars]:

$$\hat{c}_i^k = \begin{cases} 0 & \text{if } \hat{\mathbf{x}}_i^k = 0 \\ c'_i + \tilde{c}_i \hat{\mathbf{x}}_i^k + E[\tilde{f}_i] E\left[\left(\sum_{j \in J} \tilde{d}_j \hat{x}_{ij}^k - u_i\right)^+\right] & \text{otherwise} \end{cases} \quad (34)$$

where $h^+ \equiv \max\{0, h\}$. Note that, up to this point, we only require independence of excess-demand penalties and demands.

4.1. Normally distributed demands

For the special-case model, we assume that demands are independent and normally distributed with means m_j and variances v_j , i.e., $\tilde{d}_j \sim N(m_j, v_j)$. For simplicity, we also assume that all means and variances are integral. The reader will see that our techniques easily extend to means $\alpha_j m_j$ and variances $\beta_j v_j$, where α_j and β_j are positive scale parameters, and m_j and v_j represent integers running from 0 to some finite upper bounds. Given the efficiency of the dynamic-programming solution procedure that requires m_j and v_j , the scale parameters can be quite small. Thus, a wide range of actual mean-and-variance combinations can be closely approximated.

The RMP for this model is identical to the column-oriented formulation, CSFLP, presented in Section 3.2. To solve this special case exactly, we will exactly solve the

Table 5. Total time (TT) in CPU seconds, for randomly generated SFLPs with scenario uncertainty. This table explores the computational limits of our current B&P implementation with duals stabilization

Number of Scenarios	Problem Size (facilities-customers)			
	10–60	20–60	15–80	20–100
50	36.4	64.6	47.2	137.2
50	35.2	55.7	65.1	257.2
50	26.9	58.1	54.6	108.2
50	35.9	62.1	44.0	106.0
50	26.7	53.4	106.6	154.9
100	48.2	136.8	132.7	229.6
100	68.4	96.5	108.0	829.6
100	56.2	109.4	238.3	165.7
100	73.0	123.9	75.6	150.0
100	51.0	90.4	76.3	257.0
200	134.6	245.3	181.5	513.0
200	131.3	199.4	302.2	1294.6
200	103.7	168.9	762.1	260.7
200	134.0	184.5	157.1	272.2
200	103.8	174.4	156.5	555.0
300	154.8	374.4	305.4	507.8
300	248.7	305.8	391.2	817.0
300	144.6	250.0	654.1	535.4
300	164.0	320.9	493.2	493.2
300	210.8	276.0	274.4	687.1

subproblems corresponding to the formulation (29)–(32). Given Equation (34), the subproblem associated with facility i is this “static stochastic knapsack problem” (Kleywegt et al. [34]):

$$z_i^* = \min_{x_{ij} \in \{0,1\} \forall j \in J} \left\{ \sum_{j \in J} (\bar{c}_{ij} - \pi_j) x_{ij} + E[\tilde{f}_i] E \left[\left(\sum_{j \in J} \tilde{d}_j x_{ij} - u_i \right)^+ \right] \right\} + c_i - \mu_i. \quad (35)$$

Evaluating $E \left[\left(\sum_{j \in J} \tilde{d}_j x_{ij} - u_i \right)^+ \right]$ is easy, because we know that

$$E[\tilde{w}(m, v)^+] = m \Phi \left(\frac{m}{\sqrt{v}} \right) + \sqrt{\frac{v}{2\pi}} \exp \left(-\frac{m^2}{2v} \right), \quad (36)$$

for any $\tilde{w}(m, v) \sim N(m, v)$, where $\Phi(\bullet)$ denotes the cumulative distribution function of a standard normal random variable (see [34]).

To derive a complete solution, we place an arbitrary ordering on the elements of J , i.e., $J = \{1, \dots, j, \dots, |J|\}$, temporarily ignore constraint-violation penalties, and apply dynamic programming to evaluate the functions $g_i(j, m, v)$ defined here:

$$g_i(j, m, v) = \min_{x_{i1}, \dots, x_{ij}} \sum_{j'=1}^j (\bar{c}_{ij'} - \pi_{j'}) x_{ij'} \quad (37)$$

$$\text{s.t.} \quad \sum_{j'=1}^j m_{j'} x_{ij'} \leq m \quad (38)$$

$$\sum_{j'=1}^j v_{j'} x_{ij'} = v \quad (39)$$

$$x_{ij'} \in \{0, 1\} \quad \text{for } j' = 1, \dots, j, \quad (40)$$

for $m = 0, \dots, m_{\max}$, and $v = 0, \dots, v_{\max}$, where $m_{\max} = \sum_{j \in J} m_j$ and $v_{\max} = \sum_{j \in J} v_j$. (In practice, much smaller limits on m_{\max} and v_{\max} can and should be used for efficiency's sake.) The dynamic program is defined by:

Initial Conditions:

$$\begin{aligned} g_i(j, m, v) &= 0 \quad \text{for } j = 0, m = 0, v = 0, \text{ and} \\ g_i(j, m, v) &= +\infty \quad \text{for } j = 0, m \neq 0, v \neq 0. \end{aligned}$$

Recursion:

$$g_i(j, m, v) = \min \{g_i(j-1, m, v), \bar{c}_{ij} - \pi_j + g_i(j-1, m - m_j, v - v_j)\} \quad (41)$$

for $j = 1, \dots, |J|$, $m = 1, \dots, m_{\max}$, $v = 1, \dots, v_{\max}$.

This recursion is similar to that for a two-dimensional knapsack problem, but for a given m , the objective value g_i does not depend on the variance index v . This will be used in a final calculation, however.

Now, since the joint assignment $\hat{\mathbf{x}}_i$ yields an aggregate demand with distribution $N\left(\sum_{j \in J} m_j \hat{x}_{ij}, \sum_{j \in J} v_j \hat{x}_{ij}\right)$, the optimal objective value for (35) will be

$$z_i^* = \min_{\substack{m \in \{1, \dots, m_{\max}\}, \\ v \in \{1, \dots, v_{\max}\}}} \left\{ g(|J|, m, v) + E[\tilde{f}_i] E[\tilde{w}(m - u_i, v)^+] \right\} + c_i - \mu_i. \quad (42)$$

For the case in which facility capacities \tilde{u}_i are also independent and normally distributed, and independent from the customer demands, the method just described will work after making a single modification: The expectation $E[\tilde{w}(m - u_i, v)^+]$ in Equation (42) changes to $E[\tilde{w}(m - E[\tilde{u}_i], v + \text{Var}(\tilde{u}_i))^+]$.

4.2. Extensions

The methods described above will work for problems with different types of independent random demand parameters, although numerical integration may be required. Suppose, for instance, that each demand can be defined by $\tilde{d}_j = \sum_{h=1}^{H_j} \tilde{r}_{jh} + m_j$ where H_j is a modest-sized integer parameter, all \tilde{r}_{jh} are independent and identically distributed

(iid) with mean 0 and variance v , and all m_j are positive integers. Then, any aggregate demand $\sum_{j \in J} \tilde{d}_j x_{ij}$ can be described through its integral mean $m' = \sum_{j \in J} m_j x_{ij}$, and its scaled, integral variance $v' = v \sum_{j \in J} H_j x_{ij}$. Thus, assuming deterministic facility capacities u_i , the recursion (41) applies with appropriate adjustments for the scaled variances, and $E \left[\left(\sum_{j \in J} \tilde{d}_j x_{ij} - u_i \right)^+ \right]$ can be computed by numerical integration over the distribution of $\sum_{j \in J} \tilde{d}_j x_{ij}$. This will be straightforward because that distribution is defined through the mean-shifted convolution of $t = \sum_{j \in J} x_{ij}$ iid random variables, which is completely defined by its bounded integral mean m' , its bounded, scaled, integral variance v' , and the common distribution for \tilde{r}_{jh} .

4.3. Computational results for normally distributed demands

To build test instances here, we select each of the single-scenario problems from Section 3, assume costs and penalties are the deterministic quantities specified by that problem, and assume that each demand in the problem represents the expected value of an independent, normally distributed demand. The variance for each demand is then generated as a discrete uniform random variable on $[1, V_j]$, where V_j is the maximum value that assures $P(\tilde{d}_j < 0) \leq 0.001$ (e.g., Spoerl and Wood [54]). Table 6 displays solution times and integrality gaps for all 40 problem instances. We use the software suite of Section 3 for solving these problems, but the computer is an IBM G40, Pentium IV laptop computer with 1 GB of RAM, running at 3 GHz.

4.4. Discussion

As in the Section 3, we see that duals stabilization is a useful enhancement to the B&P algorithm.

With some exceptions, results displayed in Tables 1, 2, 5 and 6 indicate that B&P performs better on problems with a smaller customers-to-facilities ratio. Similar results have been observed when solving generalized-assignment problems, where the tasks-to-agents ratio correlates positively with the number of feasible solutions the problem instances have (Savelsbergh [49], Silva [52]). In turn, the number of feasible solutions correlates positively with the number of columns in the column-oriented model, and the more columns a problem has, the harder it must be to solve using B&P. The glaring contradiction to this argument comes from the 10–60 column in Table 2, which indicates that these problems are easier than the 10–50 problems. Theoretically, the 10–60 problems may have more columns than the 10–50 problems, but the effective number, i.e., the number of “cost-effective columns” may be smaller. Note that Table 4 shows that the extensive 10–60 models have tighter LP relaxations than do their 10–50 counterparts, which implies the system is more capacity-bound in some sense (which is just an artifact of the problem generator). This may imply that the only cost-effective columns are those that use most of a facility’s nominal capacity, which are relatively few in number.

Table 6. Total solution time (TT, in CPU seconds) to solve CSFLP with B&P when demands are independent, normal random variables. Each time in the bold font indicates the fastest solution time for the given problem. (Pentium IV, 3 GHz computer with 1 GB of RAM.)

Problem Size		B&P w/o Duals Stabilization			B&P with Duals Stabilization			Int. Gap (%)
Facilities	Customers	Soln. Time (TT, sec.)	Cols.	Nodes	Soln. Time (TT, sec.)	Cols.	Nodes	
5	15	0.8	188	1	0.4	194	13	0.00
		1.0	153	1	0.5	181	1	0.00
		0.5	171	1	0.4	143	1	0.00
		0.6	192	1	0.7	205	1	0.00
		0.7	168	1	0.6	191	1	0.00
5	30	11.3	443	1	9.7	404	1	0.00
		9.3	474	1	7.3	437	1	0.00
		9.3	432	1	8.8	438	1	0.01
		9.7	423	1	8.4	416	1	0.00
		18.8	599	9	17.5	695	7	0.04
8	24	1.2	350	1	1.1	337	1	0.00
		1.1	322	1	0.9	310	1	0.00
		3.0	466	9	1.7	326	3	0.05
		1.2	351	1	0.8	296	1	0.00
		1.8	385	1	1.6	363	1	0.00
8	48	9.1	1001	1	6.4	778	1	0.00
		10.2	1006	1	7.0	765	1	0.01
		11.2	1166	5	11.5	1237	9	0.02
		10.4	989	3	7.3	816	9	0.01
		11.5	959	3	7.9	734	3	0.00
10	30	3.3	512	5	2.9	455	7	0.01
		2.8	501	1	2.3	452	1	0.02
		2.7	524	1	2.0	451	1	0.00
		3.3	526	1	2.4	461	1	0.01
		2.5	497	3	5.7	856	3	0.03
10	40	6.5	823	1	5.0	708	3	0.00
		9.3	860	1	6.5	679	1	0.00
		13.9	793	3	11.5	668	3	0.02
		14.0	893	13	19.8	1326	5	0.03
		10.0	791	1	6.7	662	3	0.01
10	50	28.5	1076	1	20.7	862	3	0.03
		37.5	1180	5	22.9	794	5	0.01
		48.9	1624	17	65.3	2144	19	0.07
		32.2	1303	5	34.1	1423	5	0.02
		24.5	1105	1	17.9	837	1	0.00
10	60	73.4	1357	1	56.1	1221	1	0.00
		76.1	1585	15	46.2	1263	5	0.01
		71.7	1434	1	50.1	1213	1	0.00
		90.7	1581	5	66.9	1243	5	0.00
		74.0	1418	1	55.6	1215	1	0.00

5. Summary, conclusions and recommendations

5.1. Summary

This paper proposes a column-oriented model for a class of two-stage stochastic mixed-integer programs (SMIPs), and describes examples of well-known deterministic optimization problems whose stochastic versions fall into this class. We show how to solve such problems with a branch-and-price algorithm (B&P), using a stochastic facility-location problem (SFLP) as a computational example. We solve one version with scenario uncertainty as well as one with independent, normally distributed parameters. We solve both

Table 7. Larger instances of SFLP. Total solution time (TT, in CPU seconds) to solve CSFLP with B&P using duals stabilization, when demands are independent normal random variables. An asterisk indicates the problem does not solve in 2,400 seconds. In this case, “Cols.” is the number of columns generated in 2,400 seconds. (Pentium IV, 3 GHz computer with 1 GB of RAM.)

Problem Size (facilities-customers)											
10–60		20–60		15–80		20–100		25–150		30–200	
Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time	Soln. Time
(TT, sec.)	Cols.	(TT, sec.)	Cols.	(TT, sec.)	Cols.	(TT, sec.)	Cols.	(TT, sec.)	Cols.	(TT, sec.)	Cols.
56.1	1221	9.4	1039	110.9	1947	180.2	2421	323.2	3753	1799.7	5859
46.2	1263	9.3	1068	80.5	1719	253.6	3144	451.5	4051	2386.7	6776
50.1	1213	72.9	4092	73.8	1683	117.8	2284	1014.4	5057	*	(6644)
66.9	1243	11.6	1080	91.4	2023	153.0	2355	613.6	4025	*	(6569)
55.6	1215	29.4	2525	65.6	1585	208.9	2631	1406.7	5358	2337.7	6359

versions exactly, and demonstrate how the algorithm’s performance can be improved by “duals stabilization.” The open-source code libraries of the COmputational INfrastructure for Operations Research (COIN-OR) provide the framework for our B&P algorithm, while CPLEX 8.0 comprises the solver engine.

5.2. Conclusions

This research demonstrates that B&P is an attractive and accessible method to solve certain SMIPs. For SFLP with scenario uncertainty, B&P can be orders of magnitude faster than solving the original problem by branch and bound, and this can be true even for deterministic, i.e., single-scenario problems. Furthermore, the ability to solve exactly an SMIP with continuously distributed parameters is highly unusual in the stochastic-programming literature.

5.3. Recommendations for further work

The B&P approach can be used to solve, at least approximately, SMIPs of the class described in Section 2, but with more general probability distributions. For instance, the methods of “sample-average approximations” (Mak et al. [43], Kleywegt et al. [34]) provide probabilistic guarantees on solution quality and are based on repeated solutions of sampled approximating problems. However, a sampled approximating problem is essentially identical to a stochastic program with scenario uncertainty.

Sampled subproblems can be used to identify favorable columns in a “nearly exact algorithm,” too. Suppose that once a subproblem’s integer variables are fixed, i.e., a column of the model has been defined, the expected cost of that column can be estimated highly accurately through sampling. This certainly holds for SFLP, where the penalties associated with, say, 10,000 sampled demands for some fixed customers-to-facility assignment can be sampled and averaged in a fraction of a second. For all intents and purposes then, that average will exactly equal the expected cost for the column, and the linear-programming relaxation of a master problem containing such columns would yield exact dual solutions. The only theoretical concern in this procedure is that the solution of a sampled subproblem might indicate that a column is favorable, but extended

sampling would reveal that it is not. If we solve many sampled subproblems and cannot identify an improving column, then we might become convinced that we have, in fact, solved the LP-RMP. However, a formal procedure will need to be constructed to provide a rigorous “level of conviction.”

Finally, we note that the COIN/BCP software is originally intended for use in a distributed/parallel environment. It will be interesting to investigate how well our procedures perform in such an environment.

References

1. Ahmed, S., Sahinidis, N.V.: An approximation scheme for stochastic integer programs arising in capacity expansion. *Oper. Res.* **51**, 461–471 (2003)
2. Ahuja, R. K., Magnanti, T.L., Orlin, J.B.: *Network Flows*. Prentice Hall, Englewood Cliffs, NJ, 1993
3. Appleget, J. A., Wood, R.K.: Explicit-constraint branching for solving mixed-integer programs. M. Laguna, J.L. González-Velarde, (eds), *Computing Tools for Modeling, Optimization and Simulation*, Kluwer Academic Publishers, Boston, MA, 243–261 2000
4. Appelgren, L.H.: A column generation algorithm for a ship scheduling problem. *Transp. Sci.* **3**, 53–68 (1969)
5. Barcelo, J., Casanova, J.: A heuristic lagrangean algorithm for the capacitated plant location problem. *Eur. J. Oper. Res.* **15**, 212–226 (1984)
6. Barnhart, C., Hane, C.A., Vance, P.H.: Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.* **48**, 318–326 (2000)
7. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **46**, 316–329 (1998)
8. Beale, E.M.L.: On minimizing a convex function subject to linear inequalities. *J. R. Stat. Soc.* **17B**, 173–184 (1955)
9. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4**, 238–252 (1962)
10. Bertsimas, D.J.: A vehicle routing problem with stochastic demand. *Oper. Res.* **40**, 574–585 (1992)
11. Birge, J.R., Louveaux, F.: *Introduction to Stochastic Programming*. Springer-Verlag, New York, (1997)
12. Brown, G.G., Graves, G.: Real-time dispatch of petroleum tank trucks. *Manage. Sci.* **27**, 19–32 (1981)
13. Brown, G.G., Graves, G., Honczarenko, M.: Design and operation of a multicommodity production/distribution system using primal goal decomposition. *Manage. Sci.* **33**, 1469–1480 (1987)
14. Butchers, E.R., Day, P.R., Goldie, A.P., Miller, S., Meyer, J.A., Ryan, D.M., Scott, A.C., Wallace, C.A.: Optimized crew scheduling at Air New Zealand. *Interfaces* **31**, 30–56 (2001)
15. Butler, J.C., Dyer, J.S.: Optimizing natural gas flows with linear programming and scenarios. *Dec. Sci.* **30**, 563–580 (1999)
16. Camm J.D., Chorman, T.E., Dill, F.A., Evans, J.R., Sweeney, D.J., Wegryn, G.W.: Blending OR/MS, judgment, and GIS: Restructuring P&G’s supply chain. *Interfaces* **27**, 128–142 (1997)
17. Carøe, C.C., Tind, J.: L-shaped decomposition of two-stage stochastic programs with integer recourse. *Math. Program.* **83**, 451–464 (1998)
18. Chen, Z.-L., Li, S., Tirupati, D.: A scenario-based stochastic programming approach for technology and capacity planning. *Comput. Oper. Res.* **29**, 781–806 (2002)
19. Chu P.C., Beasley, J.E.: A genetic algorithm for the generalized assignment problem. *Comput. Oper. Res.* **24**, 17–23 (1997)
20. COIN. 2004. <http://www.coin-or.org> (accessed July 2004)
21. Damodaran, P., Wilhelm, W.E.: Branch-and-price methods for prescribing profitable upgrades of high-technology products with stochastic demands. *Dec. Sci.* **35**, 55–82 (2004)
22. Dantzig, G.B., Wolfe, P.: The decomposition principle for linear programs. *Oper. Res.* **8**, 101–111 (1960)
23. Day, P.R., Ryan, D.M.: Flight attendant rostering for short-haul airline operations. *Oper. Res.* **45**, 649–661 (1997)
24. Desrochers, M., Desrosiers, J., Solomon, M.: 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.* **40**, 342–354.
25. Desrochers, M., Solomon, F.M.: A column generation approach to the urban transit crew scheduling problem. *Transp. Sci.* **23**, 1–13 (1989)
26. Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F.: Time constrained routing and scheduling. (M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser (eds), *Handbooks in Operations Research and Management Science*, Vol. 8, Network Routing, Elsevier, Amsterdam, 35–140 (1995))

27. Du Merle, O., Villeneuve, D., Desrosiers, J., Hansen, P.: Stabilized column generation. *Discrete Math.* **194**, 229–237 (1999)
28. Ehrgott, M., Ryan, D.M.: Constructing robust crew schedules with bicriteria optimization. *Journal of Multicriteria Decision Analysis* **11**, 139–150 (2002)
29. Ford, L.R., Fulkerson, D.R.: A suggested computation for the maximal multicommodity network flows. *Manage. Sci.* **5**, 97–101 (1958)
30. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting stock problem. *Oper. Res.* **9**, 849–859 (1961)
31. Girard, A., Sansó, B.: Multicommodity flow models, failure propagation, and reliable loss network design. *IEEE/ACM Transactions on Networking* **6**, 82–93 (1998)
32. ILOG 2002. ILOG CPLEX 8.0 Reference Manual
33. Johnson, E.L.: Modeling and strong linear programs for mixed integer programming. S.W. Wallace (ed), *Algorithms and Model Formulations in Mathematical Programming*, Springer-Verlag, 1–43 (1989)
34. Kleywegt A.J., Shapiro, A., Homem-de-Mello, T.: The sample average approximation method for stochastic discrete optimization. *SIAM J. Optim.* **12**, 479–502 (2002)
35. Laporte, G., Louveaux, F.V.: The integer L-shaped method for stochastic integer programs with complete resource. *Oper. Res. Lett.* **13**, 133–142 (1993)
36. Laporte, G., Louveaux, F.V., Van Hamme, L.: Exact solution to a location problem with stochastic demands. *Transp. Sci.* **28**, 95–103 (1994)
37. Linderoth, J., Wright, S.J.: Decomposition algorithms for stochastic programming on a computational grid. Optimization Technical Report 02–07, Computer Sciences Department, University of Wisconsin-Madison (2002)
38. Lorena, L.A.N., Senne, E.L.F.: A column generation approach to capacitated p-median problems. *Comput. Oper. Res.* **31**, 863–876 (2004)
39. Lougee-Heimer, R.: The Common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM J. Res. Dev.* **47**, 57–66 (2003)
40. Louveaux, F.V., Peeters, D.: A dual-based procedure for a stochastic facility location. *Oper. Res.* **40**, 564–573 (1992)
41. Lübbecke, M.E., Desrosiers, J.: Selected topics in column generation. *Les Cahiers de GERAD G-2002-64*, Group for research in decision analysis, montreal, Canada. http://www.optimizationonline.org/DB_FILE/2002/12/580.pdf (accessed July 2004)
42. Lulli, G., Sen, S.: A branch-and-price algorithm for multi-stage stochastic integer programming with application to stochastic batch-size problems. *Manage. Sci.* **50**, 786–796 (2004)
43. Mak, W.-K., Morton, D.P., Wood, R.K.: Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Oper. Res. Lett.* **24**, 47–56 (1999)
44. Papagiannaki, K., Moon, S., Fraleigh, C., Thiran, P., Diot, C.: Measurement and analysis of single-hop delay on an IP backbone network. *IEEE Journal on Selected Areas in Communications* **21**, 908–921 (2003)
45. Ralphs, T.K., Ladanyi, L.: COIN/BCP User's Manual. <http://www-124.ibm.com/developerworks/open-source/coin/presentations/bcp-man.pdf> (accessed July 2004) (2001)
46. Ribeiro C.C., Soumis, F.: A column generation approach to the multiple-depot vehicle scheduling problem. *Oper. Res.* **42**, 41–52 (1994)
47. Rockafellar, R.T., Wets, R.J.-B.: Stochastic convex programming: relatively complete recourse and induced feasibility. *SIAM J. Control Optimization* **14**, 547–589 (1976)
48. Ryan, D.M., Foster, B.A.: An integer programming approach to scheduling. A. Wren, (ed), *Computer Scheduling of Public Transport, Urban Passenger Vehicle and Crew Scheduling*. North Holland, Amsterdam, 269–280 (1981)
49. Savelsbergh, M.W.P.: A branch-and-price algorithm for the generalized assignment problem. *Oper. Res.* **45**, 831–841 (1997)
50. Sen, S., Higle, J.L.: The C^3 theorem and a D^2 algorithm for large scale stochastic integer programming: Set convexification. *Stochastic Programming E-Print Series*. (<http://dochoost.rz.hu-berlin.de/speps>) (2000)
51. Shiina, T., Birge, J.R.: Stochastic unit commitment problem. *Int. Trans. Oper. Res.* **11**, 19–32 (2004)
52. Silva, E.F.: Improving branch-and-price algorithms and applying them to stochastic programs. PhD Dissertation, Naval Postgraduate School, Monterey, CA (2004)
53. Singh, K., Philpott, A., Wood, R.K.: Column generation for design of survivable networks. Working paper, Dept. of Engineering Science, University of Auckland, Auckland, New Zealand (2005)
54. Spoerl, D., Wood, R.K.: A stochastic generalized assignment problem. *INFORMS Annual Meeting*, Atlanta, GA, 19–22 October (2003)
55. Teo, C-P., Shu, J.: Warehouse-retailer network design problem. *Oper. Res.* **52**, 396–408 (2004)
56. Vance, P.H., Barnhart, C., Johnson, E.L., Nemhauser, G.L.: Airline crew scheduling: A new formulation and decomposition algorithm. *Oper. Res.* **45**, 188–200 (1997)

57. Walkup, D.W., Wets, R.J.-B.: Stochastic programs with recourse. *SIAM J. Appl. Math.* **15**, 1299–1314 (1967)
58. Weingartner, H.M., Ness, D.N.: Methods for the multidimensional 0/1 knapsack problem. *Oper. Res.* **15**, 83–103 (1967)
59. Wets, R.J.-B. 1966. Programming under uncertainty: the complete problem. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* **4**, 316–339 (1966)
60. Wolsey, L.A.: *Integer Programming*. John Wiley & Sons, New York, 1998
61. Zhou, J., Liu, B.: New stochastic models for capacitated location-allocation problem. *Computers and Industrial Engineering* **45**, 111–125 (2003)